An Approach to Effects Based Modeling for Wargaming

William E. Mckeever Jr., Duane A. Gilmour, Robert G. Hillman
mckeeverw@rl.af.mil, gilmourd@rl.af.mil, hillmanr@rl.af.mil

Air Force Research Laboratory, Information Directorate, Advanced Computing Technology Branch, 26 Electronics Parkway, Rome, NY 13441-4514

ABSTRACT

Effects-based operations (EBO) are proving to be a vital part of current concepts of operations in military missions and consequently need to be an integral part of current generation wargames. EBO focuses on the producing effects from military activities, as opposed to the direct result of attacking targets. Alternatively, the emphasis of conventional wargames is focused on attrition-based modeling and is incapable of assessing effects and their contribution to the overall mission objectives. For wargames to be effective, they must allow users to evaluate multiple ways to accomplish the same goal with a combination of direct, indirect and cascading events (actions). The focus of this paper is to describe the development of a methodology for the implementation of EBO concepts into modern wargames. The design approach was to develop a generic methodology and demonstrate how simulation objects can incorporate EBO capabilities. The authors will illustrate the application of the methodology utilizing an EBO scenario example, which was developed to test the system.

Keywords: effects based operations; Wargaming; military operations; center of gravity models

1. INTRODUCTION

Analysis systems are needed in the Military planning process to anticipate and respond in real-time to a dynamically changing battlespace with counter actions. Complex technical challenges exist in developing automated processes to derive hypotheses about future alternatives. Combat operations are conducted in the presence of uncertainty related to the disposition and alternatives that might emerge as events unfold. It is virtually impossible to identify or predict the specific details of what might transpire. Our research interest is to develop techniques to assess specific planned courses of actions (COAs) against the adversarial environment. Utilizing High Performance Computer (HPC) clusters, multiple force structure simulations can be executed in parallel to concurrently evaluate the hypothesis of assessing a given COA against a range of adversarial eCOAs [1]. The desired goal is to establish a means to evaluate the COA for critical elements related to execution and timing as well as overall effectiveness in the presence of a range of adversarial possibilities that may occur.

This paper addresses the technical challenge of supporting wargame simulations that assess modern campaign approaches using EBO. Conventional wargaming simulations typically execute a pre-scripted sequence of events for an adversary independent of the opposing force, commonly referred to as the blue force. In addition, conventional wargames focus on attrition force-on-force modeling, whereas modern campaign strategies are a mixture of kinetic and non-kinetic operations. One such campaign approach being pursued within the Air Force is EBO. Conventional wargames are incapable of assessing an effects-based campaign approach.

EBO is an approach to planning, executing, and assessing military operations that focuses on the effects produced from military activities, as opposed to the direct result of attacking targets [2]. EBO incorporates and expands upon traditional approaches such as targets-based and strategy-to-task. A significant challenge for EBO is predicting and assessing how blue force actions result in adversary behavioral outcomes, and how those behavioral outcomes impact the adversary commander's decisions and future actions. For wargame simulations to be effective in an effects based arena, they must allow users to evaluate multiple ways to accomplish the same goal with a combination of direct, indirect and cascading events (actions). Wargames must also be dynamic, in that the opposing force will react to blue

Enabling Technologies for Simulation Science VIII, edited by Dawn A. Trevisani, Alex F. Sisti, Proceedings of SPIE Vol. 5423 (SPIE, Bellingham, WA, 2004)
0277-786X/04/$15 · doi: 10.1117/12.543438

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUL 2006	Conference Paper Postprint	04/13/2004 - 04/15/2004

4. TITLE AND SUBTITLE

AN APPROACH TO EFFECTS BASED MODELING FOR WARGAMING

5a. CONTRACT NUMBER

In-house

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

62702F

6. AUTHOR(S)

William E. McKeever, Duane A. Gilmour & Robert G. Hillman

5d. PROJECT NUMBER

RTCO

5e. TASK NUMBER

A0

5f. WORK UNIT NUMBER

04

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

AFRL/IFTC
525 Brooks Rd
Rome NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER

AFRL-IF-RS-TP-2006-4

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFTC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

AFRL-IF-RS-TP-2006-4

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited. PA#04-169

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.
Paper presented at the SPIE Enabling Technologies for Simulation Science VIII Conference, April 13 - 15, 2004 in Orlando FL.

14. ABSTRACT

Effects-based operations (EBO) are proving to be a vital part of current concepts of operations in military missions and consequently need to be an integral part of current generation wargames. EBO focuses on the producing effects from military activities, as opposed to the direct result of attacking targets. Alternatively, the emphasis of conventional wargames is focused on attrition-based modeling and is incapable of assessing effects and their contribution to the overall mission objectives. For wargames to be effective, they must allow users to evaluate multiple ways to accomplish the same goal with a combination of direct, indirect and cascading events (actions). The focus of this paper is to describe the development of a methodology for the implementation of EBO concepts into modern wargames. The design approach was to develop a generic methodology and demonstrate how simulation objects can incorporate EBO capabilities. The authors will illustrate the application of the methodology utilizing an EBO scenario example, which was developed to test the system.

15. SUBJECT TERMS

effects based operations; wargaming; military operations; center of gravity models

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Duane A. Gilmour
U	U	U	UL	12	19b. TELEPHONE NUMBER *(Include area code)*

force actions in an unscripted manner. This paper will describe the development of a generic methodology for the implementation of EBO concepts into modern wargames, as well as demonstrate how simulation objects can incorporate the EBO capabilities. This generic methodology concept can be used for implementation of operational-level wargaming, taking into account EBO as well as attrition-based concepts.

2. OBJECTIVE

The objective of this research is to establish a modeling methodology that will demonstrate the techniques and methods to include EBO operational concepts into virtually any event based wargame simulator. Therefore, the approach is to establish a generalized methodology that could be reasonably integrated into any applicable wargame simulation. A secondary issue was to develop a modeling methodology that will allow a user to create an arbitrary EBO center of gravity (COG) [3] model that will be transformed into a simulation model for execution. The goal is to develop a single generic EBO simulation object capable of mimicking this arbitrary EBO behavior.

The following is a list of EBO modeling concepts that the system must be capable of emulating to establish an effective COA/eCOA analysis framework:

- Arbitrary input attributes that can influence or contribute to the overall system state.
- The ability to target in some manner each attribute for some type of disruption or destruction.
- Attribute items can provide observable indicators that appropriately identify the direct or indirect effect achieved.
- The ability for effects to be cascaded between multiple EBO model objects.
- The ability to support complex effects where multiple effects are used to achieved the overall objective.

To verify that the generic methodology could be utilized for the implementation of EBO concepts into modern wargames, an EBO demonstration scenario was utilized to test the system. We chose a portion of an AFRL/IF notional scenario, Operation DENY FORCE, which was designed to support a test case that highlights technologies developed by AFRL. The scenario, which will be further discussed in Section 4, highlights application of kinetic and non-kinetic operations, including Psychological Operations (PSYOPS), relating to achieving and maintaining Air Superiority within a campaign plan. It should be reiterated that the scenario is notional, and the purpose was to demonstrate how the modeling methodology could be applied to build an EBO COG model and create simulation objects representative of EBO capabilities for modern wargames.

The development process was divided into two phases. The first phase was to develop the modeling and simulation methodology as a stand alone capability and the second phase was to integrate the modeling system within a specific wargame simulator. This paper discusses the first phase of the project. We selected the Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES) framework [4] for developing the underlying simulation models. This was a somewhat arbitrary selection process based on our desire in the second phase of the project to integrate this EBO modeling methodology within our Force Structure Simulation (FSS) research project [5].

3. IMPLEMENTATION

The desire to simulate EBO characteristics as a behavioral object within a wargame simulator requires the conventional attrition-based simulation framework to be extended in several ways. The simulation must be capable of simulating not only the standard direct kinetic events but also non-kinetic actions and indirect cascading events and complex interactions that control the state of specific COGs. Finally, a method to observe the simulation objects and obtain indicators related to the state of the EBO object is required. A Java tool, referred to as "JavaCOG," was developed to support the creation EBO COG models and their related properties, developing an abstract COA and producing the parameter file that the simulation requires. This section will discuss the construction of the EBO object, the event structure, how the simulation responds to the events, and the JavaCOG tool.

3.1 Simulation Framework

The simulation framework was developed using C++ and the SPEEDES API [6]. Every EBO simulation object is created using the same base object class. In using this generic class approach, any object can exhibit EBO characteristics. The simulation object properties are initialized and constructed by parsing an input parameter file. This parameter file contains the properties that define the simulation object including name, types of actions, the action's properties, and the other EBO objects it depends on, including their cascading event properties. A cascading event is defined here as any property of a dependent node that will have an effect on the object. The EBO object creates its own specific state structure by creating an array of possible actions and properties associated with those actions. The simulation object also obtains the names of objects it relies upon; it receives events from those objects using the objects name as a trigger string. The object then creates an array of all the dependant objects along with possible cascading event properties. Object properties will be discussed in more detail below.

3.1.1 Event Framework

There are several types of events in the SPEEDES framework that were used in the simulation. This section will explain what types of events were used, why they were selected and how the events function. The first type of event is a point to point event where a simulation object schedules an event on another specific object. This event scheme is used to schedule the COA actions on an object or scan an object for an indicator. The next event type is used to allow an object to schedule an event on itself and control its internal state, for example, powering off a generator as fuel is exhausted. These events are referred to as local events. When a simulation object is affected by a cascading event or a COA event, the object calculates future actions/reactions and schedules an event on itself to notify when the new state begins.

Since the design approach taken was to have a dynamically coupled network, it's essential for an object to schedule events on multiple unknown objects. This final event type used for implementing cascading events is much more complex. A cascading event is scheduled on objects that rely on an affected object and the relation matrix for scheduling those events are unknown and arbitrarily complex. SPEEDES undirected event is well suited to accomplish this task. Undirected events are scheduled using a trigger string. The event is broadcast to all simulation objects subscribed to the specific trigger string. Therefore, when an objects state change creates a cascading event, the object will schedule an event with its name as the trigger string and the properties affected as the data. The restriction implied here is that each EBO objects must have a unique name.

3.1.2 Event Processing

The simulation objects must react to different types of events in different ways. As stated above there are many types of events that are necessary to simulate EBO object behavior. This section illustrates how the simulation object will react when it receives any of the events.

When a simulation object receives a COA event, the object identifies the type of action and then finds the properties associated with that COA action. The simulation object then calculates the impact of the action using a random number and the states probability properties to determine if the COA will have an effect on the object. If the COA fails to produce an effect, the object continues operating normally. But, if the COA action produces an effect, the simulation object updates its potential action list and schedules two local events. The first event scheduled controls the amount of time delay between the COA action and when the objects state may change, the effect. The scheduled time for the affected event is calculated by acquiring the current time then adding the delay time of the influencing action. The second local event is scheduled for when the simulation object would recover from the direct action. This event is scheduled by adding the recovery time to the current time plus the delay time.

The EBO object assesses its potential state change by testing to determine if any events were received that would cause this new state change to be ignored. There are two reasons that could cause the state to be ignored. The first is that the affect has already occurred and the object is already in that state or a new influencing factor eliminates the need for a state change. For example, if the backup system was repaired before the delay time occurred then the object would

remain operational. When all checks have been verified, the object schedules a cascading event to all of the objects that have subscribed to its trigger string to inform those objects that it has been affected. When the simulation object receives a cascading event, the EBO object repeats the same analysis and behavior processes that were performed when the COA event was detected, as described above. This type of event ripples through the simulation from object to object.

The ability of a simulation object to recover from an affected state is an important feature that was also implemented. An object can attempt to recover from an affected state after it receives one out of two events. The first type of event is a local event that was scheduled on the object itself after it has been affected by a direct action. The second event is a cascading event that signals the elimination of the indirect effect event that may change the objects current state. These two different recovery methods are both handled in virtually the same manner, and the object updates its array to reflect the repair. To find the status of the object, the object evaluates its 'objects and actions' arrays looking for an affected object in the object array or an affected action in the action array. If all of the objects and actions are normal, then the object can change its state from affected to operational. However, if there is an object still affecting it, the main object will reevaluate its own state to determine if it is still being affected. If the object can operate without the affected sub-object, then the state changes. If it is still affected by an object or an action, then the current state will remain. If the simulation objects operational status has changed, then it must produce an event to inform the objects that depend on it of the new status. The cascading recovery event is accomplished in a similar manner as the cascading event.

The last event type received by an object is a direct event to scan/query the EBO object in an attempt to observe the objects indicators. The simulation processes a scan action by printing indicators depending on the state. However, before it prints out an indicator, it will check if the indicator can be observed. Certain indicators can only be observed between certain hours of the day. The EBO object will try to print its indicator, and then evaluate all of the property values for the objects it depends on, by checking if they are operational, and checking to see if the indicator (either operating or influenced) can be printed out. By printing the indicators in this manner, it is possible to get mixed indicators and therefore, more realistic indicators. When the EBO object models are integrated into an actual wargame, the printing functions will be replaced with an event structure to respond to the object that initiated the scan.

3.2 JavaCOG – Java COG authoring tool

JavaCOG is a Java-based authoring tool that has three main functions. It serves as a graphical way to develop COG models and extends the models to include EBO characteristics. The authoring tool can be used to define an abstract COA to use in analyzing the COG system developed. The last function is to generate the parameter file for that is required to build and simulate the COG system and apply the abstract COA for testing and analysis.

JavaCOG is comprised of three windows, as seen in Figure 1. The window on the right, the main window, is the graphical way to represent the model. The window on the left is a tree representation of the nodes and the possible actions that can be scheduled or actions that have been scheduled. The bottom window is a summary of the current abstract COA in a table format with additional debugging columns.

Properties are edited in the main window. Some properties are predefined as cascading properties. The cascading property is defined in the relationship between the nodes and the directions of the arrows. The node at the tail of the arrow relies directly on the node at the head of the arrow. In Figure 1, the Power Grid relies directly on the Power Plant. If the Power Plant is affected, then the Power Grid could be affected by a cascading event.

The user can further define a node's properties by utilizing the editing mode. Figure 2 is an example of the screen a user would see when editing the Work Force node. The top half of this screen allows the user to change the name of the node, edit the actions, and edit the indicators for this node. On the lower half of the screen are the properties of all the nodes that can affect the Work Force. The Influencing Attribute drop down menu allows the user to select the influencing node to edit. The user can then select the probability that this influencing node will cause the main node to be affected. For example, in Figure 2, if the Transportation Infrastructure node was affected, there would be a high probability that the Work Force node would be affected in a cascading event. Currently, the probabilities values are set from 0.2 (very low) to 1.0 (very high), in increments of 0.2. Next, the user has the option to assign a complex effect for the influencing node. A complex effect is defined where all of the influencing nodes must be affected to influence the

main node. This translates to in this example, for the Work Force to be influenced; the Transportation Infrastructure and another node must be influenced. In this example, there is no complex effect for the Work Force node. The notional Air Superiority example shown in Section 4 will illustrate complex effects. The next property that can be edited is the delay time. The delay time is the amount of time it takes for the action to directly influence the main node. The last properties to be edited are the indicators. The indicators in this section are in reference to the observations that can be made at the main node (Work Force) that would indicate the state of the influencing node (Transportation Infrastructure).

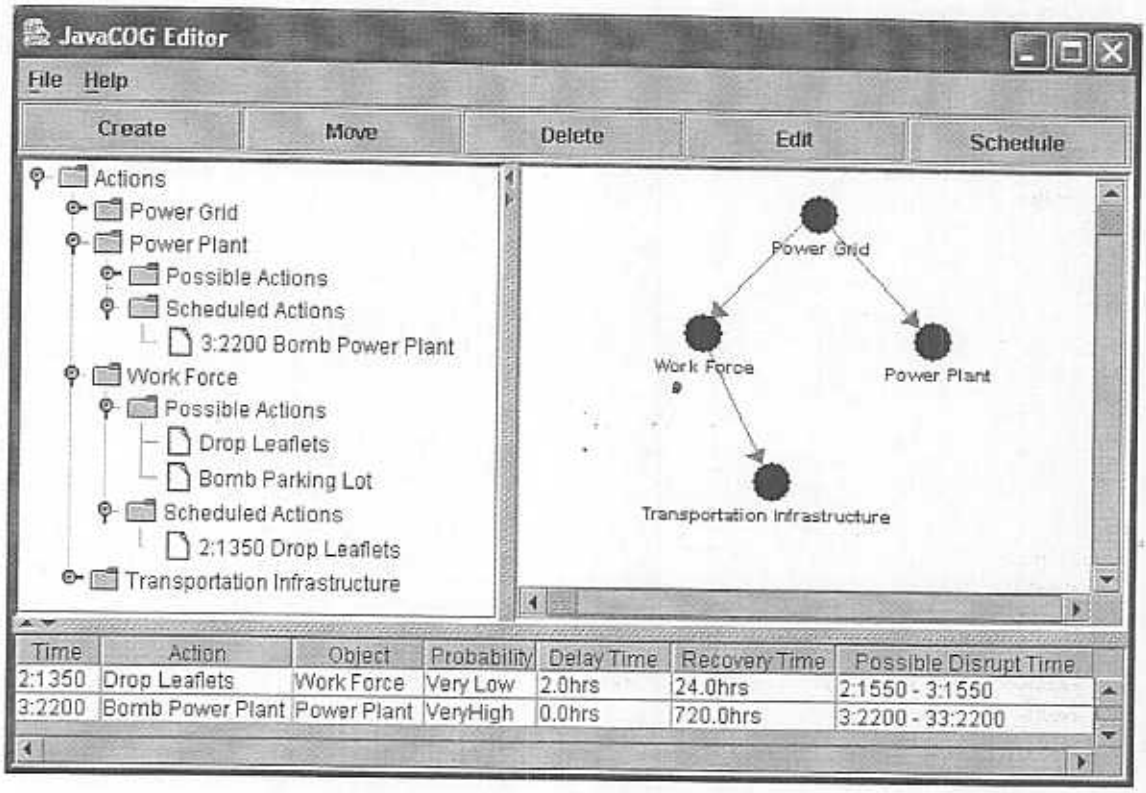

Figure 1. Example of JavaCOG

A fundamental feature of EBO is to influence the enemy by using kinetic and non-kinetic tactics. An object must be able to react to different actions in different ways. Therefore, an object must know what type of actions can be taken against it, and how to react. When modifying or creating an action, the user will have a window similar to Figure 3. From this window the user can enter the properties that the simulation will need, such as the name of the action, the delay time, the probability of success, and the amount of time it will it take for the object to recover from this action.

After the model has been created, an abstract COA set is developed with JavaCOG. The abstract COA is defined for testing purposes and therefore contains sufficient information to test the modeling system. The abstract COA consists of the intended targeting action, when it will be applied and the object that will be targeted. The table in the bottom window of Figure 1 of the JavaCOG screen represents a COA for this example. The table is comprised of seven columns. The first three columns contain the COA information described above, and the last four columns assist the user when creating the COA test set. All of the information contained within this table was input during the model creation stage, with the exception of the "Time". The "Time" column is the time when the action is scheduled. The time format is day:time, therefore 2:1350 is read as on day 2 at 1350. The next two columns represent the action to take, and the node for which the action is scheduled. The first action scheduled in the COA shown in Figure 1 is to drop leaflets on the Work Force node on day 2 at 1350.

As stated above, the next four columns serve to assist the user in viewing key GOC attributes when creating the COA test set. The probability column displays the likelihood of success of that action on the given node. The next two

columns refer to the delay and recovery time for the scheduled action on the node. The last column represents the actual time the node would be affected, if the action was successful. The first row of the table in Figure 1 can be read as follows: on day 2 at 1350 a drop leaflets action will be performed on the Work Force node, it will have a very low probability of successfully influencing the Work Force, but if it does succeed there will be a two hour delay time, the Work Force will recover 24 hours later so the possible affected time is from day 2 at 1550 to day 3 at 1550.

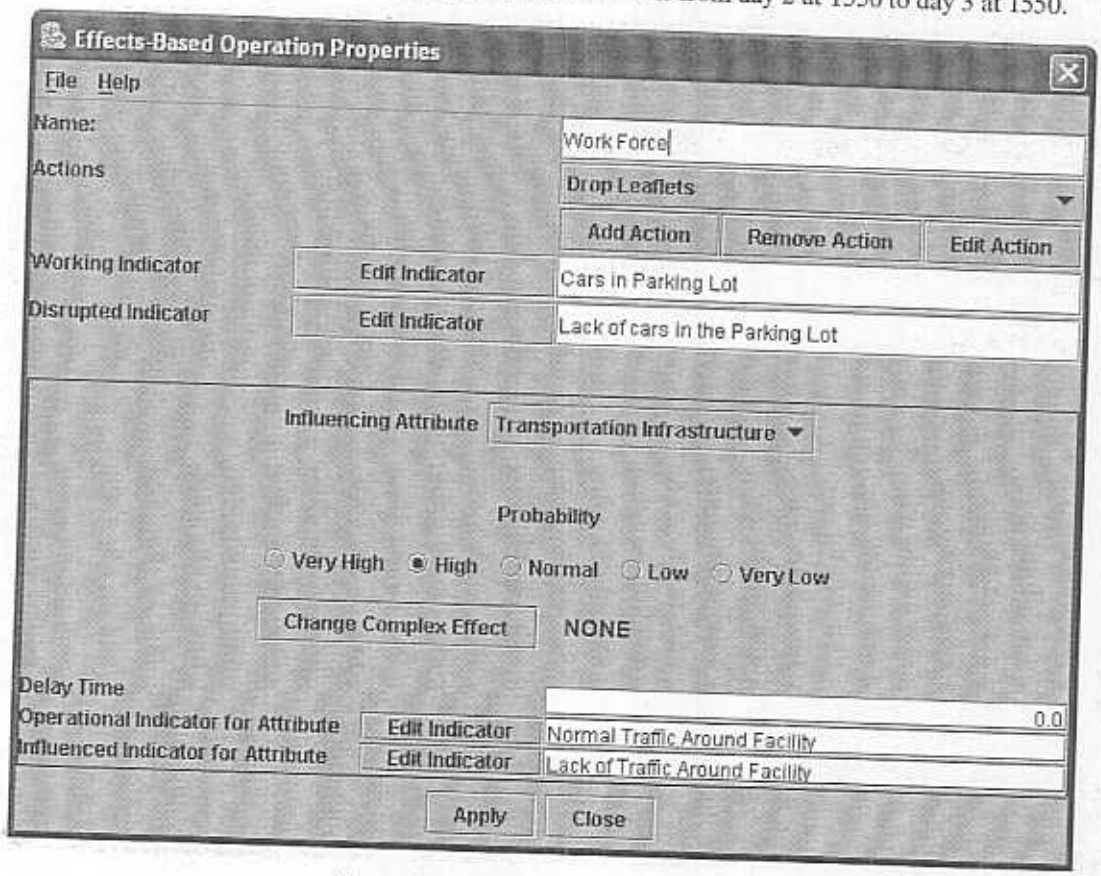

Figure 2. Example of Editing Work Force

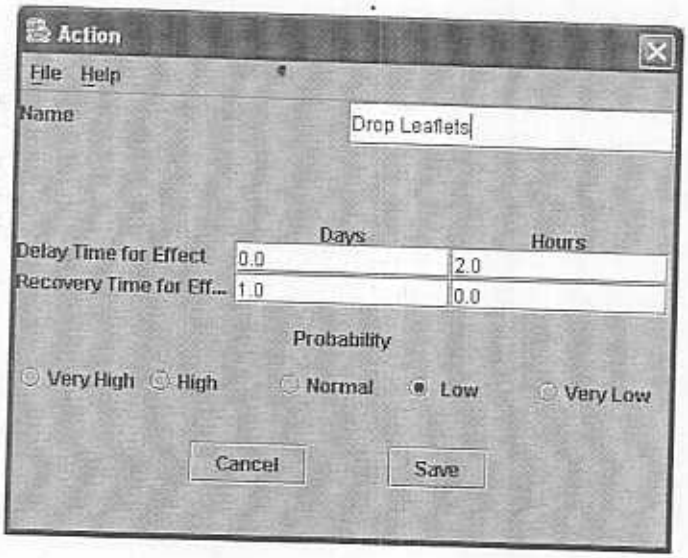

Figure 3. Example of Action Properties

Once the COG model is complete and a COA test set has been developed, the next step is to simulate the COA. The JavaCOG tool has been developed to generate a SPEEDES parameter file which can be utilized within the simulation.

4. RESULTS

To verify the operational goals and that the generic methodology was achieved, numerous test sets were constructed and analyzed during development. The final analysis process was to utilize an existing documented EBO demonstration scenario to test the system. We chose an Air Superiority scenario, which highlights the application of kinetic and non-kinetic operations, including PSYOPS. In addition to the various types of operations, numerous types of effects have been implemented, including direct, indirect, complex, cumulative and cascading [7].

4.1 EBO COG Model

The Air Superiority COG model is shown in Figure 4. This model is far from complete, but is complex enough to demonstrate the validity of the methodology and that the system can be applied in more complex EBO scenarios. The COG model was developed, such that for the blue force to achieve Air Superiority: the C2, Radar Sites and SAM Sites must all be inoperative. As you can see in Figure 4, each of these nodes is reliant on other nodes, as depicted by the head of the arrows. The way to interpret the COG model is as follows: the C2 node depends on the computer system and the Radar Sites to perform adequately; the Radar Sites depend on the Generators, Radar Comm and the Power Grid, and so forth. However, all of these dependencies should not be interpreted as a complex effect in our model. Case in point, the Radar Sites depend on a power source (Generators or Power Grid) to operate and not necessarily Comm. Similarly, the Power Grid requires either the Power Plant or the Backup Power Grid for operation, and not necessarily the Work Force.

To further define the types of EBO effects within our model, a brief explanation follows. An example of a direct effect would be bombing the Generators to make them inoperative. An indirect effect of this action would be the impact on the Radar Sites. However, as described previously, the Radar Sites would still be operational until the Power Grid is affected, hence a complex effect. A cumulative effect can be defined as follows: taking a direct action on the Power Plant and the Backup Power Grid will result in the Power Grid being inoperative. This will also result in the SAM Production Sites being inoperative, which will eventually lead to no SAMs, which results in the SAM Sites being inoperative. A cascading effect can be described in this scenario as the end result of taking direct actions over time on a number of the nodes below the C2, Radar Sites and SAM Sites resulting in the blue force Air Superiority. Once the COG model was completed with dependencies between nodes, along with applicable data, a COA was developed.

4.2 COA Development

The COA test set is outlined in Table 1, which is a screenshot of the lower portion of the main JavaCOG window for this COA. Refer to Section 3.2 for further details concerning the specific categories of Table 1. The goal of the blue force COA is to achieve Air Superiority. To accomplish this, because of the complex effect nature of the COG model, the C2, Radar Sites and SAM Sites need to be rendered inoperative. The order in which the actions were accomplished is reflected in the Table. The goal was to; first take down the Radar Sites, followed by the SAM Sites, and finally the C2 node. The timings were such that the COA lasted for about a day and a half. Leaflets were dropped on day 1 at 0400 to affect the Work Force followed by dropping a bomb on the Generators at 0430. The next action is to scan the Radar Sites indicator to determine if the desired effect has been achieved. If still operational, the next action in the COA would be to take out the Power Plant followed by the High Power Lines that connect the Backup Power Grid to the Power Grid. If successful, the Radar Sites would now be inoperative.

The next goal is to render the SAM Sites inoperative. Assuming the blue force was successful in taking down the Power Grid, the SAM Production Site would already be inoperative. Destroying the POL for SAMs would eliminate the threat of SAMs. The last node to impact blue force Air Superiority is the adversary C2 node. The network is targeted to cause confusion and the inability to order in back up forces. The last action is to scan the Air Superiority indicator to see if it has, in fact, been achieved. The COG is complete, the COA has been fully developed, and now the COA will be saved as a SPEEDES parameter file for input to the simulation tool.

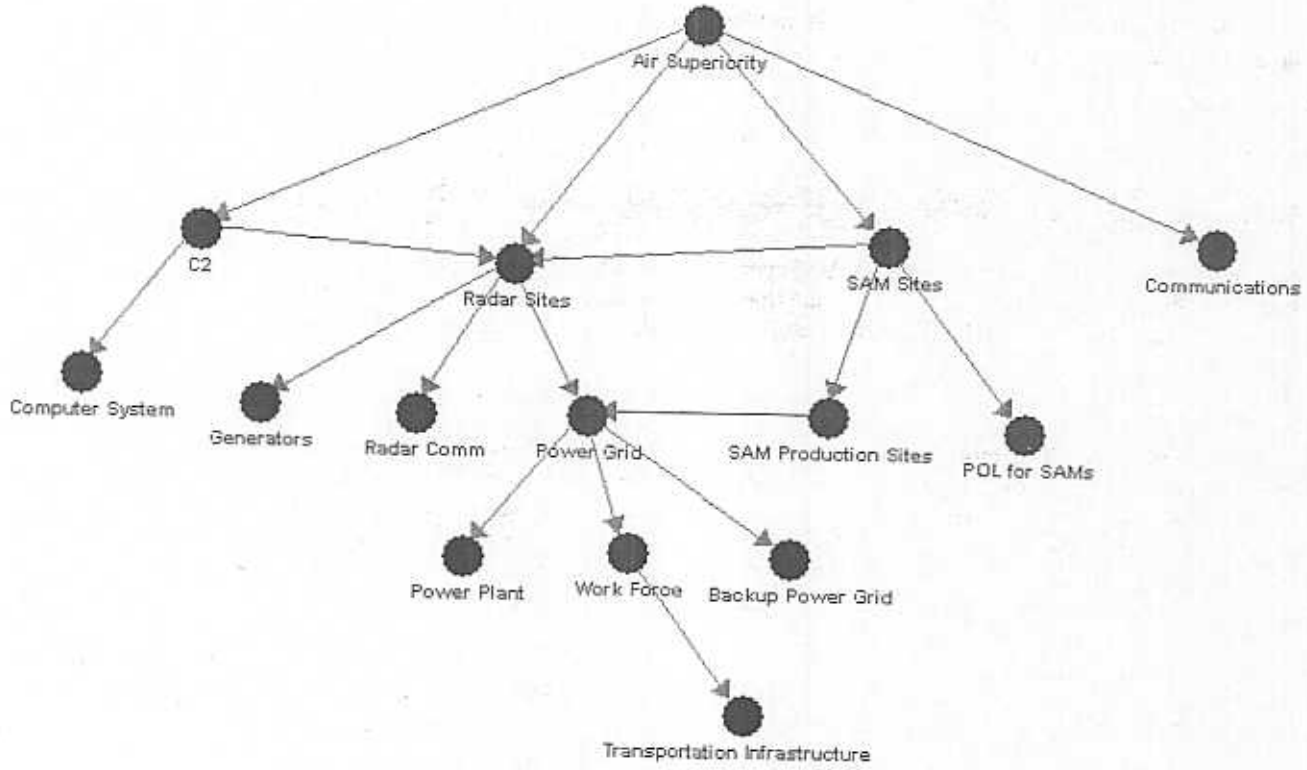

Figure 4. Air Superiority COG Model

Time	Action	Object
1:0400	Drop Leaflets	Work Force
1:0430	Bomb Generator	Generators
1:1200	Scan	Radar Sites
1:2330	Bomb Power Plant	Power Plant
1:2330	Destroy High Power Lines	Backup Power Grid
2:0030	Scan	Radar Sites
2:0200	Destroy POL for SAMs	POL for SAMs
2:0300	Take down Network	Computer System
2:0800	Scan	C2
2:1000	Scan	Air Superiority

Table 1. Air Superiority COA

4.3 COA Analysis

After running the simulation in brief reporting mode with the parameter file generated from the Air Superiority data set, the results were manually verified and can be viewed in Figure 5. In brief reporting mode, the only output displayed is when an object is scanned for indicators; this is accomplished by scheduling a corresponding COA action. The first observation detects electromagnetic waves, which is the indicator for an operational radar site. This indicates that one of the COA actions was unsuccessful at affecting their object. The next three observations show the COA actions were successful in affecting the Radar Sites, C2, and Air Superiority objects, respectively.

To further analyze the results and understand the behavior within the COG, the simulation was also run in verbose mode. The output is shown in Figure 6. In this debugging mode, the simulation will display information explaining why the object was or was not affected. The simulation prints out events; results of COA actions, recovery actions and

cascading events, as well as displaying the complex effects. The first two lines show the possible results of a COA action. The first line explains that the Dropping of Leaflets on the Work Force was unable to affect that object. The next line indicates that the Bomb Generator action on the Generators was successful. The third line in Figure 6 demonstrates a direct and complex effect that results in the Power Grid object remaining unaffected. While lines 8–19 display more cumulative and complex effects, with both successful and unsuccessful results. First, two COA actions occurred, one on the Power Plant and the other on the Backup Power Grid, then the simulation displays the cumulative effects on the Power Grid that ripple on to the Radar Sites and SAM Production Sites. Followed by line 15 where the simulation displays that the C2 object is still operating after the radar sites have been affected.

```
Scanning Radar Sites on day 1 at 12:00
    Electromagnetic Waves

Scanning Radar Sites on day 2 at 00:30
    No Electromagnetic Waves

Scanning C2 on day 2 at 08:00
    Inter-node messages less then 50% normal

Scanning Air Superiority on day 2 at 10:00
    US aircraft has unfettered access
```

Figure 5. Simulation Output

```
Work Force is operating after Drop Leaflets on day 1 at 04:00
Generators is affected by Bomb Generator on day 1 at 04:30
Generators is down but Power Grid is still operating keeping Radar Sites running

Scanning Radar Sites on day 1 at 12:00
    Electromagnetic Waves

Power Plant is affected by Bomb Power Plant on day 1 at 23:30
Power Plant is down but Backup Power Grid is still operating keeping Power Grid running
Backup Power Grid is affected by Destroy High Power Lines on day 1 at 23:30
Power Grid is affected by Backup Power Grid disruption on day 1 at 23:30
Radar Sites is affected by Power Grid disruption on day 1 at 23:30
SAM Production Sites is affected by Power Grid on day 1 at 23:30
Radar Sites is down but C2 is still operating keeping Air Superiority running
C2 is still operating without Radar Sites

SAM Sites is still operating without Radar Sites

SAM Production Sites is down but POL for SAMs is still operating keeping Sam Sites running

Scanning Radar Sites on day 2 at 00:30
    No Electromagnetic Waves

Computer System is affected by Take down Network on day 2 at 03:00
C2 is affected by Computer System on day 2 at 03:00
C2 is down but Sam Sites is still operating keeping Air Superiority running...
...
Repairing Backup Power Grid's Destroy High Power Lines on day 2 at 23:30
Backup Power Grid is operating again on day 2 at 23:30
Repairing Power Grid's Backup Power Grid on day 2 at 23:30
Power Grid is operating again on day 2 at 23:30
Repairing Radar Sites's Power Grid on day 2 at 23:30
...
```

Figure 6. Simulation Output in Debug Mode

In this example the goal of Air Superiority was achieved on day 2 at 0800 and lasted until the repair event of the High Power Lines of the Backup Power Grid on day 2 at 1130. The beginning of the repair event is displayed on the bottom of Figure 6. After the repair of the Backup Power Grid, the first repair event is sent to the Power Grid, which recovers and sends a repair event to the Radar Sites and the SAM Production Sites. These two objects then recover and send their own repair events.

5. FUTURE WORK

The final changes are being made to the EBO modeling system to expand the variability of the EBO model behavior. Currently the probability enumeration types selected identify distinct values and the simulation run is deterministic based on those initial values. The final step is to replace the value selected with a range that is evaluated with a random function during simulation. Modification of random seed values will create unique simulation results. The existing enumeration values will be replaced with an expanded set of overlapping range structures to truly emulate the uncertainly of events that may occur.

Once the implementation concept is confirmed, the next phase of the project, as mentioned previously, is to integrate the system with an existing wargame simulator. There are two approaches possible to achieve the integration. First, a High Level Architecture (HLA) [8] interface can be developed that will allow any HLA compliant simulation framework to utilize the developed modeling system. The second approach is to integrate the EBO modeling system directly into a compatible simulation framework. We selected the latter approach to integrate directly with our SPEEDES based research wargame, FSS. The code will be archived and made available to anyone that would like to develop the HLA implementation.

The integration of the EBO modeling system into the wargame simulator will require three modifications to the evaluation system developed in phase one. First, the EBO effects generator will be removed from the system or modified to support analysis rather than defining the execution scenario. The COA scenario will be replaced with the scenario generation capability utilized with our FSS system. Second, the model data set generator will be merged and integrated with the FSS system to define and build the EBO object models. Finally, the EBO models themselves will be modified to conform to and execute with the FSS simulation framework. This capability will then be utilized within our research test-bed to support our COA/eCOA research goals.

5.1 Conclusion

The COG editor allowed us to easily create a variety of new EBO model structures and with the underlying simulation system, evaluate the EBO effects. Multiple EBO COG models and test sets were developed to test and evaluate the system. The test sets were automatically converted into a simulation data set and evaluated for accuracy and realism. The EBO simulation models lay the groundwork for integrating the modeling concept into a full featured wargame simulator. The modern wargame simulators will then support an operational level EBO COA analysis process, taking into account direct, indirect, complex, cumulative and cascading effects. The next phase of developing the integrated simulation environment is under development. The top technical challenge to the successful integration of the framework will be in creating the semantic interface that needs to be established between the EBO models and utilization of the indicators as observations by the wargame simulator.

5.2 Acknowledgements

The authors would like to acknowledge the significant contributions to this project made by Mr. Martin J. Walter and Mr. James P. Hanna (Air Force Research Laboratory, Information Directorate).

5.3 References

[1] Joshua Surman, Robert Hillman, and Eugene Santos Jr., "Adversarial Inferencing for Generating Dynamic Adversary Behavior," Proceedings of the SPIE 17th Annual International Symposium on Aerospace/Defense Sensing and Controls: AeroSense 2003, Orlando, FL, April 2003.

[2] Maris J. McCrabb, "Concept of Operations for Effects-Based Operations," Aerospace Command, Control and Intelligence, Surveillance, and Reconnaissance Center, Langley AFB VA, February 2002.

[3] Carl von Clausewitz "On War," reprint Princeton University Press, Princeton, NJ, USA.

[4] Jeff Steinman, "Scalable Parallel and Distributed Military Simulations Using the SPEEDES Framework," ELECSIM 95, 1995.

[5] "Synthetic Force Structure Simulation", AFRL-IF-TR-2002-144 Final Technical Report, June 2002

[6] "Speedes API reference Manual", Metron Inc., January 2002

[7] Maris J. McCrabb, "Explaining Effects," White Paper, Air Force Research Laboratory, Rome, NY, March 2001.

[8] Judith S. Dahmann, Katherine L. Morse, "High Level Architecture for Simulation: An Update," Second International Workshop on Distributed Interactive Simulation and Real-Time Applications, Montreal, Canada, July 1998.